UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/084,721	02/25/2002	Dimitri Gorokhovik	PHFR 010019	8258

24737 7590 02/07/2008
PHILIPS INTELLECTUAL PROPERTY & STANDARDS
P.O. BOX 3001
BRIARCLIFF MANOR, NY 10510

EXAMINER	
CASCHERA, ANTONIO A	

ART UNIT	PAPER NUMBER
2628	

MAIL DATE	DELIVERY MODE
02/07/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte DIMITRI GOROKHOVIK

Appeal 2007-2949
Application 10/084,721
Technology Center 2600

Decided: February 7, 2008

Before MAHSHID D. SAADAT, JOHN A. JEFFERY, and CARLA M. KRIVAK, *Administrative Patent Judges*.

KRIVAK, *Administrative Patent Judge*.

DECISION ON APPEAL

Appellant appeals under 35 U.S.C. § 134 (2002) from a final rejection of claims 1-8. We have jurisdiction under 35 U.S.C. § 6(b) (2002).

We affirm.

STATEMENT OF CASE

Appellant's invention is a method, device, apparatus, and computer program product that controls the display of a character based on dynamic code generation of executable code from symbolic code.

Claim 1, reproduced below, is representative of the subject matter on appeal.

> 1. A method intended for controlling the display of at least one character on an output apparatus, a summary description of said character being included in a database, said method comprising the following steps:
>
> generation of an executable code from the summary description of said character,
>
> execution of the executable code corresponding to said character so as to display the character on the output apparatus,
>
> characterized in that the step of generating the executable code comprises two substeps:
>
> a step of extracting, from the summary description of said character, a nonexecutable symbolic code defining actions for coloring in points on the output apparatus,
>
> a step of dynamic generation, from said symbolic code, of the executable code.

REFERENCES

Colletti	US 5,990,907	Nov. 23, 1999
Guha	US 6,005,588	Dec. 21, 1999

The Examiner rejected claims 1, 2, 4, 5, 7, and 8 under 35 U.S.C. §103 (a) over Guha (Ans. 3-5).[1] The Examiner also rejected claims 3 and 6 under 35 U.S.C. §103 (a) over Guha and Colletti (Ans. 5-6).

Appellant contends that Guha fails to show the limitation of extracting a nonexecutable symbolic code defining actions for coloring in points from a summary description of a character (App. Br. 4 and 8).[2]

ISSUE(S)

The issue before us is whether the Examiner erred in rejecting the claims under 35 U.S.C. § 103(a) as obvious to one of ordinary skill in the art at the time of the invention. Specifically, the issue turns on whether Guha reasonably teaches or suggests extracting nonexecutable symbolic code from the summary description of a character, as claimed.

FINDINGS OF FACT

1. Appellant's invention controls the display of at least one character on an output display by generating executable code from a summary description of the character that is included in a database and executing executable code corresponding to the character. This is done by extracting nonexecutable symbolic code from the summary description and

[1] Throughout this opinion, we refer to the most recent Answer filed December 16, 2005.

[2] Throughout this opinion, we refer to the most recent Brief filed October 14, 2005, and the Reply Brief filed February 22, 2005.

dynamically generating executable code from the symbolic code. The nonexecutable symbolic code defines actions for coloring in points on the output apparatus (Spec. 2:7-16; cl. 1).

2. The executable code is stored in a storage module (cl. 2). The storage module performs like a cache memory (Spec. 4:26-31).

3. When no executable code corresponding to a character is found in the storage module, the executable code is generated. The executable code is generated in conjunction with a database including at least one summary description of a character (Spec. 5:15-16; cl. 3).

4. Symbolic code extracted from the summary description is not a directly executable code (Spec. 6:10).

5. Symbolic coding is defined as "[e]xpressing an algorithm in coded form by using symbols and numbers that people can understand (rather than the binary numbers that computers use). All modern programming languages use symbolic coding." Webster's New World Dictionary of Computer Terms 518 (8th ed. 2000) (Reply Br. 8; Appendix B).

6. Bitmaps are defined as "[t]he representation of a video image stored in a computer's memory as a set of bits. Each picture element (pixel), corresponding to a tiny dot onscreen, is controlled by an on or off code stored as a bit (1 for on, or 0 for off) for black-and-white displays. Color and shades of gray require more information. The bit map is a grid of rows and columns of the 1s and 0s that the computer translates into pixels to display

onscreen" Webster's New World Dictionary of Computer Terms 65 (8th ed. 2000) (Reply Br. 8; Appendix B).

7. In Guha, an initialization module 202 processes character sets or collections of text characters having particular characteristics such as typeface, size and style (col. 4, ll. 33-37).

8. A renderer in Guha uses a character set 201 to generate a set of character bitmaps 204. The character bitmaps are stored in a frame buffer and used as a template to generate executable code 208 (col. 4, ll. 46-55) that is subsequently stored in RAM (col. 4, ll. 66-67; Fig. 2).

9. Code generation modules in Guha use character bitmaps to generate executable code capable of reproducing one of the bitmaps when called (col. 4, ll. 53-55).

PRINCIPLES OF LAW

"[D]uring examination proceedings, claims are given their broadest reasonable interpretation consistent with the specification." *In re Hyatt*, 211 F.3d 1367, 1372 (Fed. Cir. 2000) citing *In re Graves*, 69 F.3d 1147, 1152 (Fed. Cir.1995); *In re Etter*, 756 F.2d 852, 858 (Fed. Cir.1985) (en banc). *Autogiro Co. of Am. v. United States*, 384 F.2d 391, 396 (Ct. Cl. 1967) ("Courts can neither broaden nor narrow the claims to give the patentee something different than what he has set forth [in the claim]."); *see also Continental Paper Bag Co. v. Eastern Paper Bag Co.*,

210 U.S. 405, 419 (1908); *Cimiotti Unhairing Co. v. American Fur Ref. Co.*, 198 U.S. 399, 410 (1905).

If the Examiner's burden of establishing a prima facie case of obviousness is met, the burden then shifts to the Appellant to overcome the prima facie case with argument and/or evidence. Obviousness is then determined on the basis of the evidence as a whole and the relative persuasiveness of the arguments. *See In re Oetiker*, 977 F.2d 1443, 1445 (Fed. Cir. 1992).

As articulated by the Supreme Court in *KSR Intl. Co. v. Teleflex Inc.*, 1127 S.Ct. 1727, 1741 (2007), an obviousness analysis "need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ."

ANALYSIS

The Appellant's invention is a method, device, and apparatus for controlling the display of a character in which a summary description is included in a database. Executable code is generated from the summary description, and executable code corresponding to the character is executed. The executable code is generated by extracting from the summary description, a nonexecutable symbolic code that defines actions for coloring in points on the output apparatus, and then dynamically generating the executable code from the symbolic code (FF. 1).

Guha teaches a character set that is input to a renderer which outputs character bit maps. The character bit maps are then used to generate executable code. Thus, the renderer, in effect, "extracts" the bitmaps from the character set. Significantly, the character bitmaps are themselves nonexecutable. The code generation module uses the character bitmaps to generate executable code capable of reproducing a bitmap when called (FF. 9). The executable character generation code is stored in a RAM (FF. 8).

Claims 1, 2, 4, 5, 7, and 8

With respect to representative claim 1[3], the Examiner indicates that the character sets of Guha are equivalent to Appellant's summary description (Ans. 8-9). Guha's character sets include a collection of text characters having various fonts, etc. (FF. 7). Appellant's "Description of the Related Art" states that Guha teaches "at least one set of characters, which generally consists of an alphabet and of a certain number of acronyms of a particular size and font, is present inside a database. In this database, the characters are coded according to a character definition language, …, *thus defining a summary description of a set of characters*. The steps of generating executable code…generate[s], firstly, a bitmap for each character…This bitmap is then used [to] generate[s] executable code…" (Spec. 1:11-17) (emphasis ours). Thus, by Appellant's own description, Guha teaches a summary description that generates symbolic code (bitmaps) that then generates executable code.

As noted previously, the bitmaps of Guha are said to be functionally equivalent to the nonexecutable symbolic code (Ans. 8-9) – an interpretation that we find reasonable. Furthermore, Guha employs a renderer to extract/generate a set of character bits from the character set (FF. 8). A code generation module uses the character bitmaps to generate executable code (FF. 9). Appellant states that the "symbolic code…can not be interpreted to

[3] Appellant argues claims 1, 2, 4, 5, 7, and 8 together as a group. *See* App. Br. 7-11. Accordingly, we select claim 1 as representative. *See* 37 C.F.R. § 41.37(c)(1)(vii).

be the same as the Guha bitmap because program instructions are executed on the bitmap to determine the resultant executable code, whereas the symbolic code provides instruction for determining the executable code (Br. 9). However, by this statement, both the bitmap and the symbolic code determine the executable code, e.g., they are both "nonexecutable." Turning to the claim language, we note that representative claim 1 calls for a commensurate function, namely extracting nonexecutable symbolic code from the summary description (cl. 1, FF. 1 & 4). Thus, we find Guha reasonably suggests this limitation. Appellant's arguments to the contrary are simply not commensurate with the scope of representative claim 1.

Even the definition of the term "bitmap" provided by Appellant (FF. 6) does not state that the bit map is "executable," but rather indicates that it is a "representation" which a computer "translates," thus implying that a bit map is nonexecutable.

We also find unavailing Appellant's assertion that Guha teaches away from storing the character set as this would increase storage requirements (App. Br. 8). Guha, in Figs. 1 and 3, shows storing the executable code. Further, bitmaps are stored in a frame buffer and a RAM is used to store software instructions (FF. 8, Ans. 8).

Under *In re Hyatt*, the claims are given their broadest interpretation. In this instance, we agree with the Examiner that the bitmaps are not executable giving the term its broadest reasonable interpretation, even using the definition provided in Webster. As noted above, Appellant's arguments are not commensurate in scope with the claims on appeal. Although

Appellant contends that "the Guha bitmap…program instructions are executed on the bitmap to determine the resultant executable code, whereas the symbolic code provides instruction for determining the executable code (App. Br. 9), as stated above, the bitmaps themselves are nonexecutable and they provide instruction for determining the executable code as does Appellant's symbolic code. Since we find the Examiner's interpretation of Guha reasonable with respect to the disputed limitations of representative claim 1, Appellant has not persuasively rebutted the Examiner's prima facie case of obviousness for that claim.[4]

Dependent Claims 3 and 6

Likewise, we sustain the Examiner's rejection of dependent claims 3 and 6 under 35 U.S.C. § 103(a) as being unpatentable over the teachings of Guha and Colletti. We find that the Examiner has established at least a prima facie case of obviousness for these claims on pages 5 and 6 of the Answer which has not been persuasively rebutted. Apart from merely alleging that claims 3 and 6 are allowable by virtue of their dependence on allowable base claims (App. Br. 12), Appellant has not specifically pointed out or explained the purported deficiencies in the Examiner's position. For the foregoing reasons, the rejection is therefore sustained.

[4] If the Examiner's burden of establishing a prima facie case of obviousness is met, the burden then shifts to the Appellant to overcome the prima facie case with argument and/or evidence. Obviousness is then determined on the basis of the evidence as a whole and the relative persuasiveness of the arguments. *See In re Oetiker*, 977 F.2d 1443, 1445 (Fed. Cir. 1992).

CONCLUSION

We therefore conclude that the Examiner did not err in rejecting claims 1-8 under 35 U.S.C §103(a).

DECISION

The decision of the Examiner rejecting claims 1-8 is affirmed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED

eld

PHILIPS INTELLECTUAL PROPERTY & STANDARDS
P.O. BOX 3001
BRIARCLIFF MANOR NY 10510